"Express Mail" mailing label number EL823499921US

APPLICATION FOR LETTERS PATENT
OF THE UNITED STATES

NAME OF INVENTORS: MARK J. FLANAGAN
8735 INDIAN RIVER RUN
BOYNTON BEACH, FL 33437

TITLE OF INVENTION: INTERACTIVE COLLABORATIVE FACILITY FOR INSPECTION AND REVIEW OF SOFTWARE PRODUCTS

TO WHOM IT MAY CONCERN, THE FOLLOWING IS
A SPECIFICATION OF THE AFORESAID INVENTION

INTERACTIVE COLLABORATIVE FACILITY FOR INSPECTION AND REVIEW OF SOFTWARE PRODUCTS

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention is related to software package development and more particularly to reviewing and approving software products at various stages of development.

Background of the Invention

Most large development projects such as developing a consumer software package are collaborative efforts. Typically, these are modular projects where individuals or small groups are each assigned a specific development task. Such a task may be to write a specific piece of code for large software package, e.g., a subroutine, a macro, etc. Normally, after each participant develops his or her particular code to a point where it may be complete or close to completion, the code is reviewed for approval.

Prior to the review, the author or a moderator creates a static packet of the code for review and distributes the packet to all reviewers. After a suitable period of time, the author or moderator holds the packet review meeting wherein one person, e.g., the author or developer, reads the code aloud and reviewers are allowed to comment and identify errors or problems with the code. Comments are recorded and the author may take whatever action he or she feels is necessary such as, for example, correcting errors or rewriting portions of code. These code inspection or review meetings are integral parts of software development processes.

Generally, these code inspections are done manually, providing each reviewer with a paper copy of the code and, review/inspection meetings generally take a great deal of time. A typical code inspection/review meeting may cover 100

to 125 lines of code per hour. For a large software module, e.g., a couple of thousand lines long, this review can be a long tedious process.

SUMMARY OF THE INVENTION

It is a purpose of the present invention to reduce the time required by reviewers to review an item;

It is another purpose of the present invention to facilitate code development and review.

The present invention is a review facility and method for interactively reviewing an item, such as a block of code, by a group of reviewers and collecting comments from the reviewers. An author provides the item in text for preparation and interactive review. The text is prepared, tagging each line and making the tagged text available to reviewers, e.g., as links in a web page. Reviewers interactively comment on individual lines and review previously entered comments. An abbreviated formal review may follow the interactive review.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a flow diagram of an example of an interactive collaborative facility according to a preferred embodiment of the present invention;

Figure 2 is a flow diagram of an example of the first primary step;

Figure 3 shows a flow diagram of an example of the interactive, collaborative review process step.

DESCRIPTION OF THE PREFERRED EMBODIMENT OF THE PRESENT INVENTION

Turning now to the drawings, Figure 1 shows a flow diagram of an example of an interactive collaborative facility 100 according to a preferred embodiment of the present invention. The interactive collaborative facility 100 may be described as

including three primary phases or steps. In the first primary step 102, items (e.g., software code including, for example, modules, macros, subroutines, etc.) are prepared for consideration or review. In step 104, the items are reviewed interactively in an on-line review, e.g., by remotely located reviewers connecting to the facility over the Internet to access and review items. Participants review the items and comment on-line. Comments are collected and stored in a central repository. Then, at an abbreviated formal review meeting in step 106, any outstanding issues that may remain for any item are resolved to complete the review in 108.

It is understood that, although described herein for reviewing software, the present invention may be applied to any similar review process, such as for example, a document review wherein a master copy of a document is distributed to multiple individuals who individually annotate the document and return the annotated document to the distribution. Comments for such a document may be directed to any selected block of text or granularity, i.e., sentence, paragraph, page, chapter or any other selected text block size. Advantageously, applying the present invention to such a process accelerates final approval of the document or other project.

Figure 2 shows a flow diagram of an example of the first primary step 102 of Figure 1. In step 1022 the author retrieves the item to be prepared for review, e.g., from personal storage. Next, the item, e.g., a static code packet, is formatted for inspection in step 1024. Formatting the item for inspection defines review granularity (i.e., the specificity of review) and identifies individual elements in step 1026 (e.g., a line of source code) within the item for comments by reviewers. Then, all the identified elements are gathered and collected into a known location, e.g., a database in a central repository. Coincidentally, an initially empty comment file is constructed in step 1028 for collecting comments, questions and noting errors. The comment file also may be passed to the central repository.

So, for example, a preparation facility accepts a collection of source code, e.g., as a text file, and generates formatted HTML pages with line numbers, each linked to a comment entry form. Also, a table of contents pointing to individual items or source code blocks is prepared for the entire source code file. Then, the

prepared items and the comment file are made available for review in step 104, e.g., as an interactive web page. The web page displays, for example, the table of contents and any comments for the currently selected item. Reviewers are prompted to review the source code and review begins by selecting individual item links and entering comments on elements (lines) in the comment file. If additional file information detail is desired, e.g., source code metrics, etc., the preparation facility may provide for collecting such information. Source code metrics may include, for example, the number of lines of text and/or code, the number of comments, etc.

In yet another example, a simple Perl script file may be created (e.g., "make_web_review.pl") to accept source code and produce formatted, linked HTML pages ready for publication. The third-party (open-source) Perl package "code2html" may be used to "pretty-print" the source code, e.g., as a web page or in a frame on a web page. Code display formatting may be adjusted by modifying a code2html.css style sheet. Each source file identified to the make_web_review script is formatted into a new subdirectory, e.g., filename.ext.html. A table of contents is generated with links to each of these source files. Also, a generic review frame set and generic instructions page is copied into each subdirectory, while a javascript source file for cookie handling and report linkage is generated for each particular review. So, for a C/C++ file a code analyzer, such as the Resource Standard Metrics tool from M Squared Technologies, may be used to compute standardized metrics for the code to be inspected.

Continuing this example, the make_web_review script accepts specific parameters to locate the source files and tailor the generated package. These parameters may include a parameter specifying a base path (base) to the source files being prepared. A destination parameter (dest) specifies the path to the destination subdirectory where the prepared files are to be stored. In some systems, this destination subdirectory must exist before running the make_web_review script. A HTML link location variable (html) indicates the prepared package location and is generally different than the dest location. A HTML link location specifies the Common Gateway Interface (CGI) portion of the review facility as described below. This CGI link may be in a centralized CGI directory or a link that is unique to each review, as selected by the web administrator controlling the review publication

facility. Any remaining parameters are provided by another variable that specifies either specific file paths, paths to directories containing files, or subdirectories of the base parameter where the source files to be prepared are located.

Figure 3 shows a flow diagram of an example of the interactive collaborative review process step 104 carried out by each reviewer. This is basically an iterative process that begins in step 1040 when reviewers are prompted to review the source code. Then, for each reviewer in step 1042 the reviewer selects a linked item, e.g., the first time through step 1042 the reviewer selects the first file or item to inspect. In step 1044 the reviewer is presented with any comments that have already been entered and decides whether the item is acceptable or, if it needs further comment. Comments are linked to an element within the selected item and may be displayed for the reviewer in a separate window or frame. If any comments are required, then in step 1046, the reviewer enters his/her comments to indicate changes that should be made to one element or that another element contains an error, etc., and the comments are included with those from other reviewers.

Preferably, comments are collected in a central repository, as each is made and immediately made available to reviewers, e.g., by the reviewer refreshing his/her display. Thus, each reviewer is presented with all previous and current comments, similar to chat in a chat room or messages in an instant messenger. This reduces the likelihood of redundant comments, e.g., 15 reviewers noting that the same word in line 23 is misspelled. Optionally, all comments are collected for each reviewer and distributed together after the reviewer has completed review. After commenting or, if the reviewer decides that no additional comments are required for the current element, a check is made in step 1048 to determine whether more items (more files) remain to be reviewed. If more remain, returning to step 1042, the reviewer selects the next element and continues the review. Once all of the items have been reviewed, the reviewer's comments have been received and distributed and comments from other reviewers have been provided to the reviewer, the item may be passed for a formal review meeting in step 106.

The author or inspection moderator may provide comment responses to all reviewers. Either contemporaneously as comments are entered or, when all

reviewers have completed their review and provided their comments. Each reviewer may review the author's response, checking off those that are clear and where corrections are simple. Any remaining comments are classified as inspection meeting items or action items. By providing each reviewer with an opportunity to enter comments prior to an inspection/review meeting, the majority of corrections, especially for minor errors, are addressed prior to the meeting. The otherwise very time-consuming "reading" process is eliminated from the inspection meeting to allow meeting participants to focus on more serious items, i.e., those where significant discussion may still be necessary.

This interactive collaborative review step 104 involves two classes of activities. In the first class of activity is collaborative, where collaborators view comments and examine the item in detail as a file placed on a server using a web browser, for example. The collaborators can comment, pose questions or note errors found in the file. The comments are passed back to the server and placed in a comment file. Optionally, each user may be allowed to selectively view all remarks for the file or all remarks for a particular user. The comment file is made available interactively to all the collaborators along with an identification of both the source of each comment and the element within the item (source code file and line number in this example) to which the particular comment is directed. The second class of activity is file management and communication activity by the server managing the files with the formatted item, the item elements and the comment file. The server makes the elements and reviewer comments available to reviewers and accepts any additional individual reviewer comments and author responses, also making those available.

The preferred collaborative facility 100 of Figure 1 may include a Perl program executing as a CGI extension to an Apache web server. As reviewers enter comments into a remark entry form, the entries are passed as CGI data to the Perl program which validates the information form and records remarks in the central review data store. Also, the same CGI program may provide for retrieving previously stored information, e.g., comments, responses, etc. Optionally, at the reviewer's discretion, comment retrieval may be limited to a particular file, set of files or inspection elements, a particular user's comments, combinations thereof, or

all stored information. Each remark in the comment file is hyper-linked to the specific element (file and line number) to which it applies.

So, after the collaborative interactive review step 104, a more traditional formal review meeting is held in step 106. As noted above, this formal review meeting 106 has much fewer issues to resolve and so, typically, is much more abbreviated from what would normally have taken place because most comments have been received and acted upon by the author prior to the meeting 106. Thus, this formal review meeting 106 is limited, typically, to reviewing only changes or corrections made to the code, remarks entered and collected during the review by the collaborators and clarifying remarks that may not be understood. Once this formal review meeting 106 is complete, the item (e.g., macro, subroutine, etc.) is good to go.

Thus, the actual code inspection meeting of step 106 need only to focus on comments already provided as discussion points. This greatly reduces the actual meeting time for the inspection meeting because most of the reviewers have already reviewed the code and so, most of the code does not require additional consideration. After any outstanding issues are resolved, the author completes the inspection process by correcting the identified problems and the moderator verifies the corrections. Re-publishing the corrected code is optional and may be useful when the item is subject to corrections.

The inspection report (i.e., the output from the review meeting) may be simplified using collected metrics from the comment review file. For example, a typical well known text processing tool may be used to count the number of major and minor errors identified. Also, many companies use standardized metrics, lines of code, number of comments, etc. to measure productivity. Metrics such as the number of errors found per 1000 lines of code, number of defects detected at various times–during code reviews, during component testing, during system testing--can be useful as well. Typically, many of these metrics are used to justify various expenditures for improved tools or the like, but occasionally differing project methodologies are compared using this kind of metric. For example, the 1970's style "waterfall" methodology of development fell to the 1990's style "object

oriented" (OO) methodology based on studies that showed that the number of defects in the resulting product measured against the development costs went down using OO. So, if desired, a list of files, metrics, and comments both minor and major can be collected directly from the data.

Thus, for example output of a project, e.g., blocks of source code, is formatted and treated as a series of related web pages, one for each block. A table of contents is displayed providing a series of web links or internet links to the blocks. Selecting a link can select one file to be reviewed and the selected file is displayed. Within the displayed file links are provided at each line of code, for example. Formatting a text file to interface with viewers is well understood in the art. Once formatted, selecting a line of code brings up an interface for inputting comments about the selected line, e.g., opening a chat room, instant messenger or a frame somewhere on the page where comments are entered and displayed. Furthermore, the comments may be classified or typed, for example, as Comments, Questions, Corrections and, Major/ Minor error annotations. Thus, a reviewer can pull up a group of one particular type of comment, e.g. all questions, all corrections, etc. Also, the reviewer can selectively view comments, e.g., all comments for one reviewer, and respond. The formal review is scheduled after an appointed date, e.g., a scheduled end of review date, which may coincide with the end of the normal prior art review period, i.e., the end of the period allocated for the prior manual review.

Advantageously, because all reviewers have commented and received responses to most if not all comments, the formal review proceeds much more quickly and efficiently to a conclusion and to approval of the code. Thus, the present invention is a much more efficient and quicker process than prior art reviews.

While the invention has been described in terms of preferred embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.